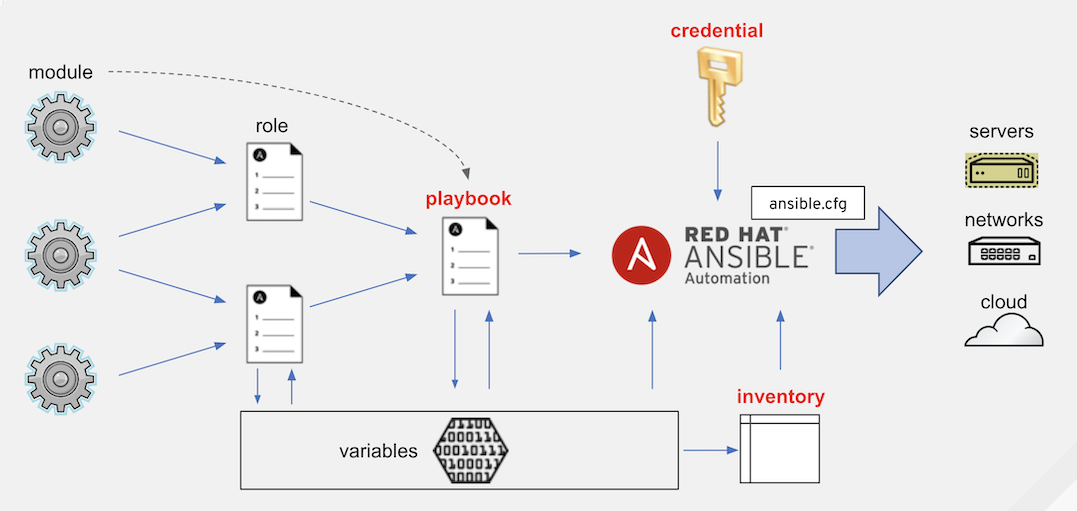
Ansible Basics, Inventory and Credentials

We will learn about the basics of Ansible: inventory and credentials. These are two of the three minimum pieces of information you need to have in order to run Ansible.



Running Ansible in an Exercise Environment

First, please run the following command. This is using Ansible to check the disk usage of the three exercise nodes.

cd ~/

ansible all -m shell -a 'df -h'

node-1 | CHANGED | rc=0 >>

Filesystem Size Used Avail Use% Mounted on

/dev/xvda1 10G 885M 9.2G 9% /

devtmpfs 473M 0 473M 0% /dev

tmpfs 495M 0 495M 0% /dev/shm

tmpfs 495M 13M 482M 3% /run

tmpfs 495M 0 495M 0% /sys/fs/cgroup

tmpfs 99M 0 99M 0% /run/user/1000

node-2 | CHANGED | rc=0 >>

Filesystem Size Used Avail Use% Mounted on

/dev/xvda1 10G 885M 9.2G 9% /

devtmpfs 473M 0 473M 0% /dev

tmpfs 495M 0 495M 0% /dev/shm

tmpfs 495M 13M 482M 3% /run

tmpfs 495M 0 495M 0% /sys/fs/cgroup

tmpfs 99M 0 99M 0% /run/user/1000

node-3 | CHANGED | rc=0 >>

Filesystem Size Used Avail Use% Mounted on

/dev/xvda1 10G 885M 9.2G 9% /

devtmpfs 473M 0 473M 0% /dev

tmpfs 495M 0 495M 0% /dev/shm

tmpfs 495M 13M 482M 3% /run

tmpfs 495M 0 495M 0% /sys/fs/cgroup

tmpfs 99M 0 99M 0% /run/user/1000

Note: Please ignore any differences between the actual output and the example output above. The important thing is that df -h has been executed

Note: df command retrieves the usage status of the disks on the server

Now we can get disk usage information from three nodes. But how were these three nodes determined? Of course, this is preconfigured for the exercise, but some of you may be wondering where that information is set in Ansible. We'll go over the settings.

ansible.cfg

First, please execute the following command.

ansible --version

ansible [core 2.11.5]

config file = /root/.ansible.cfg

configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']

ansible python module location = /opt/kata-materials/ansible/lib/python3.8/site-packages/ansible

ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections

executable location = /opt/kata-materials/ansible/bin/ansible

python version = 3.8.10 (default, Sep 28 2021, 16:10:42) [GCC 9.3.0]

jinja version = 3.0.2

libyaml = True

Note: Output content may vary depending on the environment

If you run the ansible command with the --version option, it will output some basic information about the execution environment. This includes the version and the Python version you are using. Here we will focus on the following line.

* config file = /root/.ansible.cfg

Note: If you are running the exercise on jupyter labs, it will be /jupyter/.ansible.cfg . In the following exercises, replace /root with /jupyter

This shows the path to the Ansible configuration file that will be loaded when you run the ansible command in this directory. This file is a configuration file to control the basic behavior of Ansible.

Ansible searches ansible.cfg in a specific order, though it uses the phrase "when run in this directory". Details can be found in [Ansible Configuration Settings](https://docs.ansible.com/ansible/latest/reference_appendices/config.html" \l "the-configuration-file" \t "_blank).

Briefly, ansible.cfg is searched in the following order: the path given by the environment variable ANSIBLE\_CONFIG , the current directory, the home directory, and the common path of the whole OS. In this exercise environment, ~/.ansible.cfg in the home directory is used because it is found first.

Let's check out what this contains.

cat ~/.ansible.cfg

[defaults]

inventory = /root/inventory\_file

host\_key\_checking = False

force\_color = True

[ssh\_connection]

ssh\_args = -o ControlMaster=auto -o ControlPersist=60s -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null

Several settings have been configured for the exercise. The following settings are important here.

* inventory = /root/inventory\_file

This is a setting related to the "inventory" where Ansible decides which nodes to run the automation on.

Let's take a closer look at this setting next.

Inventory

Inventory is what Ansible uses to determine which nodes to run automations on. Let's take a look at the contents of the file.

cat ~/inventory\_file

[web]

node-1 ansible\_host=3.114.16.114

node-2 ansible\_host=3.114.209.178

node-3 ansible\_host=52.195.15.8

[all:vars]

ansible\_user=centos

ansible\_ssh\_private\_key\_file=/root/aitac-automation-keypair.pem

Note: Depending on your environment, you may see output like http\_access=http://35.73.128.87:8083 , but do not worry about it and please proceed

This inventory is written in the form of an ini file. It also support other formats such as YAML and Dynamic Inventory which dynamically configure the inventory with scripts. For more details, please check [How to build your inventory](https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html).

This inventory file is described by the following rules.

* Information is described by one node per line, such as node-1 node-2 .
  + A node line consists of identifier of the node (node-1) and <span class='kc-markdown-code-copy'></span>host variable(s) to be given to the node (ansible\_host=xxxx).
  + You can also specify an IP address or FQDN for the node-1 part.
* You can create a group of hosts with [web] . Here, a group named web will be created.
  + You can use any group name except all and localhost .
    - e.g. [web] [ap] [db] is used to group the system
* In [all:vars] , group variables are defined for the group all .
  + all is a special group, a group that points to all nodes described in the inventory.
  + The ansible\_user ansible\_ssh\_private\_key\_file given here is a special variable that points to the username and SSH private key path used to login to each node.
    - A [magic variable](https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html) represented by ansible\_xxxx , which contains special values that control Ansible's behavior and environment information that Ansible will automatically retrieve. Details are explained in the variables section.

Let's try to run Ansible against a node defined using this inventory. Please run the following command.

ansible web -i ~/inventory\_file -m ping -o

node-3 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

node-1 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

node-2 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

The meanings of the options for this command are as follows

* web : Specifies a group in the inventory.
* -i ~/inventory : Specifies the inventory file to use.
* -m ping : Runs the module ping . Details about the module are described later.
* -o : Summarize the output into one line per node.

In this environment, the ansible.cfg file specifies the default inventory, so you can omit -i ~/inventory\_file as follows.

ansible web -m ping -o

node-3 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

node-1 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

node-2 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

Note: In the following exercises, we will omit the inventory specification as above

It is also possible to specify the node name instead of the group name.

ansible node-1 -m ping -o

node-1 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

It is also possible to specify multiple nodes.

ansible node-1,node-3 -m ping -o

node-1 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

node-3 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

Let's specify a special group, all . The all group covers all the nodes in the inventory. In this inventory, the all and web groups point to the same thing, so the result will be the same.

ansible all -m ping -o

node-1 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

node-2 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

node-3 | SUCCESS => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python"},"changed": false,"ping": "pong"}

In the example so far, Ansible performs some kind of execution (in this case, ping) on the specified group, but it is also possible to check only the target nodes without execution. In this case, use the --list-hosts option.

ansible web --list-hosts

hosts (3):

node-1

node-2

node-3

ansible node-2 --list-hosts

hosts (1):

node-2

Credentials

In the above inventory check, we ran the ping module on three nodes. This module actually logs in to the nodes to check if Ansible is ready to run. It checks the credentials used for login.

Note: This is completely different from the ping command which sends ICMP used in networking. We are running ping as an Ansible module.

In this exercise environment, the authentication information is specified in the inventory we checked at earlier. The following is an excerpt.

[all:vars]

ansible\_user=centos

ansible\_ssh\_private\_key\_file=/root/aitac-automation-keypair.pem

We define [all:vars] as a variable for all groups, and define the variables to be used for authentication there.

* ansible\_user : Specify the username that Ansible will use for login.
* ansible\_ssh\_private\_key\_file : Specify the private key that Ansible will use for login.

In this exercise, we use the private key, but you can also specify a password for login.

* ansible\_password : Specify The password that Ansible will use to log in.

Several other methods of giving credentials are also provided. One of the most common is to give it as a command line option.

ansible all -u centos --private-key ~/aitac-automation-keypair.pem -m ping

* -u centos : Specify the user name to use for login.
* --private-key : Specify the private key to use for login.

You can also use a password. The following is a sample.

$ ansible all -u centos -k -m ping

SSH password: ← You'll be asked to enter the password here.

node-1 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python"

},

"changed": false,

"ping": "pong"

}

node-2 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python"

},

"changed": false,

"ping": "pong"

}

node-3 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python"

},

"changed": false,

"ping": "pong"

}

Note: In this exercise environment, the private key is set in the inventory file, so the operation will succeed even if the wrong password is entered because the key authentication is given priority.

* -k : Prompts for the password when the command is executed.

There are several other ways to pass authentication information to Ansible. In this exercise, we use the most basic and convenient method (direct specification with variables). However, when you use it in production, you need to carefully consider how to handle the authentication information in advance. Of course, authentication information written directly in the file can be used by anyone who has access to the file for other purposes (such as illegally operating the server).

In general, it is often used in combination with automation platform software such as [Ansible Automation Platform](https://www.redhat.com/ja/technologies/management/ansible" \t "_blank) or [AWX](https://github.com/ansible/awx).

BACKNEXT



